[illegible]

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/886,178	06/20/2001	David Wallman	SUN1P830/P6124	5987

22434 7590 01/30/2006

BEYER WEAVER & THOMAS LLP
P.O. BOX 70250
OAKLAND, CA 94612-0250

EXAMINER
CHOW, CHIH CHING

ART UNIT	PAPER NUMBER
2192	

DATE MAILED: 01/30/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

PTO-90C (Rev. 10/03)

Office Action Summary	Application No.	Applicant(s)	
	09/886,178	WALLMAN ET AL.	
	Examiner	Art Unit	
	Chih-Ching Chow	2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.
- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *07 November 2005*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1 and 3-31* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☐ Claim(s) _____ is/are rejected.

7)☒ Claim(s) *1 and 3-31* is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *22 April 2005* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date 01/18/05 , 12/01/05
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to amendment dated November 07, 2005.

2. Per Applicants' request, independent claims 1, 11, and 20 have been amended, claim 2 canceled, and new claims 26-31 added.

3. Claims 1, 3-31 remain pending.

4. A request for continued examination under 37 CFR 1.114, including the fee set forth in 37 CFR 1.17(e), was filed in this application after final rejection. Since this application is eligible for continued examination under 37 CFR 1.114, and the fee set forth in 37 CFR 1.17(e) has been timely paid, the finality of the previous Office action has been withdrawn pursuant to 37 CFR 1.114. Applicant's submission filed on 09/02/2005 has been entered.

Double Patenting

5. The nonstatutory double patenting rejection is based on a judicially created doctrine grounded in public policy (a policy reflected in the statute) so as to prevent the unjustified or improper timewise extension of the "right to exclude" granted by a patent and to prevent possible harassment by multiple assignees. See *In re Goodman*, 11 F.3d 1046, 29 USPQ2d 2010 (Fed. Cir. 1993); *In re Longi*, 759 F.2d 887, 225 USPQ 645 (Fed. Cir. 1985); *In re Van Ornum*, 686 F.2d 937, 214 USPQ 761 (CCPA 1982); *In re Vogel*, 422 F.2d 438, 164 USPQ 619 (CCPA 1970); and, *In re Thorington*, 418 F.2d 528, 163 USPQ 644 (CCPA 1969).

A timely filed terminal disclaimer in compliance with 37 CFR 1.321(c) may be used to overcome an actual or provisional rejection based on a nonstatutory double patenting ground provided the conflicting application or patent is shown to be commonly owned with this application. See 37 CFR 1.130(b).

Effective January 1, 1994, a registered attorney or agent of record may sign a terminal disclaimer. A terminal disclaimer signed by the assignee must fully comply with 37 CFR 3.73(b).

6. Claim 1 is rejected under the judicially created doctrine of obviousness-type double patenting as being unpatentable over claim 1 of U.S. Patent No. 6,964,033. Although the conflicting claims are not identical, they are not patentably distinct from each other, from the comparison listed in the following table:

Current Application (09/886,178) US 2003/0088855A1	**US Patent No. 6,964,033**
Claim 1	Claim 1

In a **JAVA computing environment**, including a **JAVA virtual machine**, a method of generating optional attributes in a JAVA class file, said method comprising: **receiving as input JAVA runtime environment optimization information** indicating **JAVA application bytecodes that are associated with Java objects** of interest for the particular JAVA application; **generating one or more optional attributes** based on said **JAVA runtime environment** optimization information; and writing said one or more optional **attributes in an attribute table portion of a JAVA class file,** wherein said one or more optional **attributes are processed by the JAVA virtual machine** to optimize **execution of the JAVA virtual machine** for the particular JAVA application by controlling how **JAVA runtime environment** features are provided for the particular JAVA application.	In a **Java computing environment**, a method of customizing a **Java runtime environment** for a Java class file which is executed by a **virtual machine** said method comprising: reading a java class file, which includes a plurality of Bytecodes, prior to loading or executing the class file by the virtual machine; analyzing said **JAVA class file**, for one or more runtime attributes associated with runtime performance of the java class file, prior to loading or executing the class file by the virtual machine; marking one or more **Bytecodes** of said class file, based on said analyzing of said class file, prior to loading or executing the class file by the virtual machine; **generating at least one runtime attribute** for each one of said one or more marked Java Bytecodes prior to loading or executing the class file by the virtual machine; reading by the virtual machine, during the load time of the class file into the virtual machine, the at least one runtime attribute for each one of said one or more marked Java Bytecodes; loading by the virtual machine at least one runtime feature associated with the at least one runtime attribute prior to execution of said class file by the virtual machine; and executing said class file by said virtual machine in a runtime environment that includes the at least one runtime feature associated with the at least one runtime attribute, thereby allowing the runtime environment to be customized based on the analyzing, marking, and generating of the at least one runtime attribute.

Claim 1 of current application is anticipated by US Patent No. 6,964,033 claim 1 in that US Patent No. 6,964,033 claim 1 contains all the limitations of the current application claim 1. Claim 1 of the current application therefore is not patentably distinct from US Patent No. 6,964,033 claim 1 and as such is unpatentable for obvious-type double patenting.

This is an obviousness-type double patenting rejection because the conflicting claims have in fact been patented.

Claim Objections

7. Claims 1, 11, 20, 26-28, and 30-31 are objected to because of the following informalities: The use of the trademark JAVA has been noted in this application. It should be capitalized wherever it appears and be accompanied by the generic terminology. Appropriate correction is required.

Claim Rejections - 35 USC § 103

8. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

9. Claims 1- is rejected under 35 U.S.C. 103(a) as being unpatentable over Robert J. Cyran et al. U.S. Patent No. 6,412,107 (hereinafter "Cyran"), in view of US Patent 5,943,496 by Li et al. (hereinafter "Li").

CLAIM	**Cyran / Li**
1. In a JAVA computing environment, including a JAVA virtual machine, a method of generating optional attributes in a JAVA class file, said method comprising: a. receiving as input JAVA runtime environment optimization information indicating JAVA application bytecodes that are associated with Java objects of interest	For items (a) and (b), in Cyran's Abstract, "The present invention is a code preparation system (12) which **accepts input code** (11) in intermediate code format, our source code format which is first translated into intermediate format, analyzes the intermediate code, then provides **optimization information**, hints,

for the particular JAVA application;

b. generating one or more optional attributes based on said JAVA runtime environment optimization information; and

c. writing said one or more optional attributes in an attribute table portion of a JAVA class file,

d. wherein said one or more optional attributes are processed by the JAVA virtual machine to optimize execution of the JAVA virtual machine for the particular JAVA application by controlling how JAVA runtime environment features are provided for the particular JAVA application.

and/or directions (collectively referred to as '**optimization information**')". Also Cyran's column 3, lines 1-3, "it is assumed that the input code 11 is Java source code or **bytecodes**, and that the intermediate code generated by the code preparation system 12 when the input code 11 is Java source code is Java bytecodes". Further, refer to FIG.4, and see column 6, lines 16-20, "operation continues at decision block 56 where the user may choose to **perform optimization analysis** on the **input code** *(receiving input Java code)*. If **selected** (*optional*), operation continues to block 58 where the code analyzer 36 is invoked to pre-process, i.e., pre-compile the **input code.**" Further, in Cyran column 4, lines 48-55, "the presence of this **attribute in the class file** 14 informs the **Java runtime system** to JIT compile the intermediate codes for the method that includes the 'Code' **attribute** (*generating one or more optional attributes*). In addition, the COM.TexasInstruments.JIT **attribute** is also used to pass **optimization information** generated by the code preparation system 12 or directions to the JIT compiler instructing it to perform its own **optimizations at compile time**, as discussed hereinabove." – Cyran's teaching is not 'merely a reallocation of resource usage" (REMARKS dated 08/04/2005, page 6, 4th paragraph). It avoids unnecessary loading of the attribute data, see FIG. 4, the method only loads needed attributes when it's needed.

For item (c), in Cyran column 2, lines 51-54, "The **input code** 11 may also be source code, such as **Java source code**, which is first translated into intermediate code format before being analyzed. The

optimization information it provided as additional **attributes** added to **class files** 14 (*writing into attribute table*) generated by the code preparation system 12." Cyran teaches all aspects of claim 1, but he does not mention 'one or more optional attributes are processed by the JAVA virtual machine to optimize execution of the JAVA virtual machine for the particular JAVA application' specifically. However, Li teaches it in an analogous prior art. In Li's FIG. 5 and FIG. 7, Li teaches a Java computing environment which allows to create an JAVA object instance at run time and initialization of created component and view objects according to a specific attribute. For item c, see Li's FIG. 3 cutomization behavior object class name field 320 customized attribute definition block 310. Fig. 4 Customized attribute data block 414.
For item d, see Li's column 2, lines 45-58, "On the most basic level, the object model of the present invention separates certain types of data from a Java object which would otherwise be encapsulated in the object"; and Li's claim 1, "A method for specifying an object instance corresponding to an object class comprising the steps of: generating an object type information file which specifies an internal interface (*application programming interface*) for the object instance, the object type information file also containing definition data which map attribute data to memory allocated in the object instance according to the definition data, the attribute data specifying an external interface for the object instance; and **generating the attribute data** in an application file separate from the object type information

	file; wherein the object type information file and the **attribute data in the application file are employed together to create the object instance during execution of the application file.** It would have been obvious to a person of ordinary skill in the art at the time of the invention was made to supplement Cyran's disclosure of the optimization Java environment with the customization object behavior and attribute teachings by Li as noted above. And one would want to do so because this would provide the Cyran's system with a more efficient means to deal with optional (customize) attributes in a JAVA class file in a JAVA runtime environment, e.g. able to maintain attribute data typically encapsulated in an object therefore by maintaining attribute data typically encapsulated in an object separate from the object thereby realizing a number of efficiencies with regard to object instantiation. (see Li's column 1, lines 11-13, emphasis added, and at column 2, lines 20-26).
2. (Cancelled)	
3. A method as recited claim 1, wherein said method further comprises: generating computer program code that implements an application programming interface suitable for loading said one or more optional attributes.	See claim 1 rejection.
4. A method as recited claim 3, wherein said application programming interface can be used to read said one or more optional attributes from said JAVA class file.	See claim 1 rejection.
5. A method as recited claim 4, wherein	See claim 1 and 4 rejections.

said application programming interface includes functions that can be used to read first, last, and next optional attributes in said JAVA class file.	
6. A method as recited claim 4, wherein said application programming interface includes a function suitable for finding an optional attribute in said JAVA class file.	For the feature of claim 1 see claim 1 rejection. See Cyran FIG. 2, where 'storage device' can be a database, also in claim 1 (c) rejection, the attributes are stored in a 'table' which can also be a database (see Johnson FIG. 2).
7. A method as recited claim 1, wherein said JAVA runtime optimization is stored in a database.	For the feature of claim 1 see claim 1 rejection. See Cyran FIG. 1, the Extended Class File is an extension of the Code Preparation System, which is suitable for analyzing a Java application. Also see Li's FIG. 7, the 'Objet/Name Table' is a database.
8. A method as recited in claim 7, wherein said database is generated by a compiler extension or a software tool suitable for analyzing a JAVA application.	For the feature of claim 7 see claim 7 rejection. See Li column 2, lines 37-44, "an **application development tool** is described which allows application developers to easily **customize instances of common Java** object classes according to the present invention without subclassing or knowledge of the Java programming language. The virtual machine extension (VMX) of the present invention imposes a new object model on the Java object model provided intrinsically by the Java programming language."
9. A method as recited in claim 7, wherein said database is stored in a runtime performance manager that can interact with software modules that generate and load said one or more optional attributes.	For the feature of claim 7 see claim 7 rejection. See claim 8 rejection, to programming tool can interact with the software modules
10. A method as recited in claim 7, wherein	Same as claim 1 rejection; the 'code

said method further comprises: updating said database to reflect generation of said one or more optional attributes.	preparation system' disclosed in Cyran's prior art functions as an 'attribute generator'.
11. In a JAVA computing environment, including a JAVA virtual machine, a JAVA optional attribute generator computer-implemented method suitable for generation of optional attributes in a JAVA class file, said JAVA optional attribute generator computer-implemented method operating to: receive as input a JAVA runtime environment optimization information indicating JAVA application bycodes that are associated with Java objects of interest for the particular JAVA application; generate one or more optional attributes based on said JAVA runtime environment optimization information; and write said one or more optional attributes in an attribute table portion of a JAVA class file, wherein said one or more optional attributes are processed by the JAVA virtual machine to optimize execution of the Java virtual machine for the particular JAVA application by controlling how JAVA runtime environment features are provided for the particular JAVA application.	See claim 1 rejection.
12. A JAVA optional attribute generator as recited in claim 11, wherein said JAVA optional attribute generator computer-implemented method operates to generate computer program code that implements an application programming interface suitable for loading said one or more optional attributes.	For the feature of claim 11 see claim 11 rejection. For the rest of claim 12 feature see claim 3 rejection.

13. A JAVA optional attribute generator as recited in claim 11, wherein an application programming interface can be used to read said one or more optional attributes from said JAVA class file.	For the feature of claim 11 see claim 11 rejection. For the rest of claim 13 feature see claim 4 rejection.
14. A JAVA optional attribute generator computer-implemented method as recited in claim 11, wherein said JAVA runtime environment optimization information is stored in a database.	For the feature of claim 11 see claim 11 rejection. For the rest of claim 14 feature see claim 7 rejection.
15. A JAVA optional attribute generator computer-implemented method as recited in claim 11, wherein said database is generated by a compiler extension or a software tool suitable for analyzing a JAVA application.	For the feature of claim 11 see claim 11 rejection. For the rest of claim 15 feature see claim 8 rejection.
16. A JAVA optional attribute generator computer-implemented method as recited in claim 11, wherein said database is stored in a runtime performance manager that can interact with software modules that generate and load said one or more optional attributes.	For the feature of claim 11 see claim 11 rejection. For the rest of claim 16 feature see claim 9 rejection.
17. A JAVA optional attribute generator computer-implemented method as recited in claim 11, wherein said optional attribute generator operates to update said database to reflect generation of said one or more optional attributes.	For the feature of claim 11 see claim 11 rejection. For the rest of the feature see claim 10 rejection.
18. A JAVA optional attribute generator computer-implemented method as recited in claim 11, wherein said optional attribute generator operates to generate a description of an optional attribute.	For the feature of claim 11 see claim 11 rejection. In Cyran, column 8, lines 57-64, "The code preparation system 12 also passes **directions** (*description*) to the code interpretive runtime system in addition to passing optimization information. These

	directions are usually relating to optimizations that must be done on generated native code as opposed to being done on intermediate code. Thus, for example, **directions** can be given to the JIT compiler to perform **instruction** scheduling and peephole optimizations as described hereinabove in the **description** of the COM.TexasInstruments.JITOptimizations attribute."
20. A computer readable medium including computer program code for generating optional attributes in a JAVA class file for a JAVA computing environment including a JAVA virtual machine, said computer readable medium comprising: computer program code for receiving as input a JAVA runtime environment optimization information indicating JAVA application bycodes that are associated with Java objects of interest for the particular JAVA application; computer program code for generating one or more optional attributes based on said Java runtime environment optimization information; and computer program code for writing said one or more optional attributes in an attribute table portion of a Java class file wherein said one or more optional attributes are processed by the JAVA virtual machine to optimize execution of the Java virtual machine for the particular JAVA application by controlling how JAVA runtime environment features are provided for the particular JAVA application.	Same as claim 1 rejection; a readable medium is disclosed in Cyran's prior art, see FIG. 2.
21. A computer readable medium as recited in claim 20, wherein said method further	For the feature of claim 20 see claim 20 rejection. For rest of claim 21 feature see

comprises: generating computer program code that implements an application programming interface suitable for loading said one or more optional attributes.	claim 3 rejection.
22. A computer readable medium as recited in claim 21, wherein said JAVA runtime environment optimization information is stored in a database.	For the feature of claim 21 see claim 21 rejection. For rest of claim 22 features see claim 7 rejection.
23. A computer readable medium as recited in claim 22, wherein said database is generated by a compiler extension or a software tool suitable for analyzing a JAVA application.	For the feature of claim 22 see claim 22 rejection. For rest of claim 23 features see claim 8 rejection.
24. A computer readable medium as recited in claim 22, wherein said database is stored in a runtime performance manager that can interact with software modules that generate and load said one or more optional attributes.	For the feature of claim 22 see claim 22 rejection. For rest of the features see claim 9 rejection.
25. A computer readable medium as recited in claim 24, wherein said method further comprises: updating said database to reflect generation of said one or more optional attributes.	For the feature of claim 24 see claim 24 rejection. For rest of claim 25 feature see claim 10 rejection.
26. (New) The method of claim 1, wherein: the optional attributes indicate to the JAVA virtual machine which features of the Java runtime environment need to be loaded for the particular JAVA application.	For the feature of claim 1 see claim 1 rejection. For rest of claim 26 feature see Cyran's column 5, lines 15-20, "the compilation attribute informs the runtime system to JIT compile the intermediate code **for the particular method**. (*particular JAVA application*) With the additional of sub-attribute information, it may also inform the JIT compiler to **perform optimizations** such as instruction

	scheduling and peephole optimizations (*special treatment*) on the native code that it generates."
27. (New) The method of claim 1, wherein: the optional attributes indicate to the JAVA virtual machine that some Java objects are to be given special treatment at runtime.	See claim 26 rejection.
28. (New) The JAVA optional attribute generator computer-implemented method of claim 11, wherein: the optional attributes indicate to the JAVA virtual machine which features of the Java runtime environment need to be loaded for the particular JAVA application.	For the feature of claim 11 see claim 11 rejection. For rest of claim 28 feature see claim 26 rejection.
29. (New) The JAVA optional attribute generator computer-implemented method of claim 11 wherein: the optional attributes indicate to the JAVA virtual machine that some Java objects are to be given special treatment at runtime.	For the feature of claim 11 see claim 11 rejection. For rest of claim 29 feature see claim 27 rejection.
30. (New) The computer-readable medium of claim 20, wherein: the optional attributes indicate to the JAVA virtual machine which features of the Java runtime environment need to be loaded for the particular JAVA application.	For the feature of claim 20 see claim 20 rejection. For rest of claim 30 feature see claim 26 rejection.
31. (New) The computer-readable medium of claim 20, wherein: the optional attributes indicate to the JAVA virtual machine that some Java objects are to be given special treatment at runtime.	For the feature of claim 20 see claim 20 rejection. For rest of claim 31 feature see claim 27 rejection.

10. Claim 19 is rejected under 35 U.S.C. 103(a) as being unpatentable over Robert J. Cyran et al. U.S. Patent No. 6,412,107 (hereinafter "Cyran"), in view of US Patent 5,943,496 by Li et al. (hereinafter "Li"), and further in view of Cary Lee Bates et al. U.S. Patent No. 6,769,015 (hereinafter "Bates").

CLAIM	**Cyran / Li / Bates**
19. A JAVA optional attribute generator computer-implemented method as recited in claim 18, wherein said description is in XML format.	For the feature of claim 18 see claim 18 rejection. Cyran and Li teach all aspects of claim 19, but he does not mention 'in XML' specifically, however; Bates teaches it in an analogous prior art. In Bates' column 6, lines 22-23, "The attributes may be encoded using, e.g., HTML or XML (extensible markup language)". It would have been obvious to a person of ordinary skill in the art at the time of the invention was made to modify Cyran and Li's disclosure of the Attributes Generator in a Java environment by using XML taught by Bates, for the purpose of retrieving the attribute information provide more versatile JAVA system (Bates column 6, lines 21).

Conclusion

The following summarizes the status of the claims:

35 USC § 103 rejection: Claims 1, 3-31

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Chih-Ching Chow whose telephone number is 571-272-3693. The examiner can normally be reached on 7:30am - 4:00pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on 571-272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300. Any inquiry of a general nature of relating to the status of this application should be directed to the **TC2100 Group receptionist: 571-272-2100**.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Chih-Ching Chow
Examiner
Art Unit 2192
January 18, 2006

CC

[illegible]

TUAN DAM
SUPERVISORY PATENT EXAMINER